AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method to indicate to a dynamic compiler of a multitasking virtual machine when the dynamic compiler can skip generating native code for a class initialization barrier when compiling a program method, wherein a runtime representation of classes by the multitasking virtual machine includes a shared runtime data structure that is shared by multiple tasks, and wherein a native code produced by the dynamic compiler extends the shared runtime data structure representing classes and can be executed serially or concurrently by multiple tasks of the multitasking virtual machine, the method comprising:

augmenting the shared runtime data structure with an initializer field, wherein the shared runtime data structure represents a shared part of a class, and wherein the initializer field is a variable; and

using the value of the initializer field of the class to determine whether a platform-independent instruction of the program method may trigger an initialization of the class.

2. (Original) The method of claim 1, further comprising initializing a bootstrap class, wherein the bootstrap class is initialized during startup of a task of the multitasking virtual machine, and before any concurrency, due to creation of multiple threads of control within the task, takes place.

3. (Original) The method of claim 2, further comprising assigning a value of an initializer of the class when the class is fully initialized, wherein the value includes an indication that one of:

the class is not the bootstrap class,

the class is the bootstrap class and the value of the initializer identifies a holder of the class initialization barrier that triggered the initialization of the class, and

the class is the bootstrap class and the value of the initializer further indicates that the class initialization was not triggered by a class initialization barrier.

4. (Original) The method of claim 3, further comprising:

setting a binary variable to zero upon starting the multitasking virtual machine; and

setting the binary variable to one when all bootstrap classes have been initialized by a first task executed by the multitasking virtual machine;

whereby the binary variable indicates to the multitasking virtual machine whether all bootstrap classes have been initialized.

5. (Original) The method of claim 4, further comprising:

upon initiating the initialization of the class from a class initialization barrier, noting the holder of the class initialization barrier; and

once the class is fully initialized, assigning the holder to the initializer field only if the binary variable is zero.

6. (Original) The method of claim 3, further comprising upon setting a non-bootstrap class to a fully initialized state for a task, assigning the initializer

field of the class to a constant value, wherein the constant value is distinguishable from all other possible values for the initializer field.

7. (Original) The method of claim 6, wherein the constant value is a NULL pointer.

8. (Original) The method of claim 3, wherein a pointer to a runtime data structure representing the shared part of the class is assigned to the initializer field of the class to indicate that the class is the bootstrap class whose initialization is not triggered by a class initialization barrier.

9. (Original) The method of claim 3, further comprising:

if the class does not have an initialization sequence,

setting the class to a fully initialized state upon the class being loading by the task, and

assigning the initializer field of the class to a pointer to a runtime data structure representing the shared part of the class.

10. (Original) The method of claim 3, further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the class targeted by the class initialization barrier is equal to the class that defines the program method being compiled.

11. (Original) The method of claim 10, further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the class targeted by the class initialization barrier is a superclass of the class that defines the program method being compiled.

12. (Original) The method of claim 11, wherein the value of the holder of the class initialization barrier is a pointer to a runtime data structure representing the shared part of the class that defines the program method that holds the class initialization barrier.

13. (Original) The method of claim 12, further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the value of the initializer field of the class targeted by the class initialization barrier is:

different from the value that indicates that the class is not bootstrap class, and

different from the pointer to the runtime data structure representing the shared part of the class that defines the program method being compiled.

14. (Original) The method of claim 11, wherein the value of the holder of the class initialization barrier is a pointer to the shared runtime data structure representing the program method that holds the class initialization barrier.

15. (Original) The method of claim 14, further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the value of the initializer field of the class targeted by the class initialization barrier is:

different from the value that indicate that the class is not the bootstrap class, and

different from the pointer to the shared runtime data structure representing the program method being compiled.

16. (Currently amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method to indicate to a dynamic compiler of a multitasking virtual machine when the dynamic compiler can skip generating native code for a class initialization barrier when compiling a program method, wherein a runtime representation of classes by the multitasking virtual machine includes a shared runtime data structure that is shared by multiple tasks, and wherein a native code produced by the dynamic compiler extends the shared runtime data structure representing classes and can be executed serially or concurrently by multiple tasks of the multitasking virtual machine, the method comprising:

augmenting the shared runtime data structure with an initializer field, wherein the shared runtime data structure represents a shared part of a class, and wherein the initializer filed is a variable; and

using the value of the initializer field of the class to determine whether a platform-independent instruction of the program method may trigger an initialization of the class.

17. (Original) The computer-readable storage medium of claim 16,the method further comprising initializing a bootstrap class, wherein the bootstrap class is initialized during startup of a task of the multitasking virtual machine, and before any concurrency, due to creation of multiple threads of control within the task, takes place.

18. (Original) The computer-readable storage medium of claim 17, the method further comprising assigning a value of an initializer of the class when the class is fully initialized, wherein the value includes an indication that one of:

the class is not the bootstrap class,

the class is the bootstrap class and the value of the initializer identifies a holder of the class initialization barrier that triggered the initialization of the class, and

the class is the bootstrap class and the value of the initializer further indicates that the class initialization was not triggered by a class initialization barrier.

19. (Original) The computer-readable storage medium of claim 18, the method further comprising:

setting a binary variable to zero upon starting the multitasking virtual machine; and

setting the binary variable to one when all bootstrap classes have been initialized by a first task executed by the multitasking virtual machine;

whereby the binary variable indicates to the multitasking virtual machine whether all bootstrap classes have been initialized.

20. (Original) The computer-readable storage medium of claim 19, the method further comprising:

upon initiating the initialization of the class from a class initialization barrier, noting the holder of the class initialization barrier; and

once the class is fully initialized, assigning the holder to the initializer field only if the binary variable is zero.

21. (Original) The computer-readable storage medium of claim 18, the method further comprising upon setting a non-bootstrap class to a fully initialized state for a task, assigning the initializer field of the class to a constant value, wherein the constant value is distinguishable from all other possible values for the initializer field.

22. (Original) The computer-readable storage medium of claim 21, wherein the constant value is a NULL pointer.

23. (Original) The computer-readable storage medium of claim 18, wherein a pointer to a runtime data structure representing the shared part of the class is assigned to the initializer field of the class to indicate that the class is the bootstrap class whose initialization is not triggered by a class initialization barrier.

24. (Original) The computer-readable storage medium of claim 18, the method further comprising:

if the class does not have an initialization sequence,

setting the class to a fully initialized state upon the class being loading by the task, and

assigning the initializer field of the class to a pointer to a runtime data structure representing the shared part of the class.

25. (Original) The computer-readable storage medium of claim 18, the method further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the class targeted by the class initialization barrier is equal to the class that defines the program method being compiled.

26. (Original) The computer-readable storage medium of claim 25, the method further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the class targeted by the class initialization barrier is a superclass of the class that defines the program method being compiled.

27. (Original) The computer-readable storage medium of claim 26, wherein the value of the holder of the class initialization barrier is a pointer to a runtime data structure representing the shared part of the class that defines the program method that holds the class initialization barrier.

28. (Original) The computer-readable storage medium of claim 27, the method further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the value of the initializer field of the class targeted by the class initialization barrier is:

different from the value that indicates that the class is not bootstrap class, and

different from the pointer to the runtime data structure representing the shared part of the class that defines the program method being compiled.

29. (Original) The computer-readable storage medium of claim 26, wherein the value of the holder of the class initialization barrier is a pointer to the shared runtime data structure representing the program method that holds the class initialization barrier.

30. (Original) The computer-readable storage medium of claim 29, the method further comprising instructing the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the value of the initializer field of the class targeted by the class initialization barrier is:

different from the value that indicate that the class is not the bootstrap class, and

different from the pointer to the shared runtime data structure representing the program method being compiled.

31. (Currently amended) An apparatus to indicate to a dynamic compiler of a multitasking virtual machine when the dynamic compiler can skip generating native code for a class initialization barrier when compiling a program method, wherein a runtime representation of classes by the multitasking virtual machine includes a shared runtime data structure that is shared by multiple tasks, and wherein a native code produced by the dynamic compiler extends the shared runtime data structure representing classes and can be executed serially or concurrently by multiple tasks of the multitasking virtual machine, comprising:

an augmenting mechanism that is configured to augment the shared runtime data structure with an initializer field, wherein the shared runtime data structure represents a shared part of a class, and wherein the initializer field is a variable; and

a determining mechanism that is configured to use the value of the initializer field of the class to determine whether a platform-independent instruction of the program method may trigger an initialization of the class.

32. (Original) The apparatus of claim 31, further comprising an initializing mechanism that is configured to initialize a bootstrap class, wherein the bootstrap class is initialized during startup of a task of the multitasking virtual machine, and before any concurrency, due to creation of multiple threads of control within the task, takes place.

33. (Original) The apparatus of claim 32, further comprising an assigning mechanism that is configured to assign a value of an initializer of the class when the class is fully initialized, wherein the value includes an indication that one of:

the class is not the bootstrap class,

the class is the bootstrap class and the value of the initializer identifies a holder of the class initialization barrier that triggered the initialization of the class, and

the class is the bootstrap class and the value of the initializer further indicates that the class initialization was not triggered by a class initialization barrier.

34. (Original) The apparatus of claim 33, further comprising:

a setting mechanism that is configured to set a binary variable to zero upon starting the multitasking virtual machine;

wherein the setting mechanism is further configured to set the binary variable to one when all bootstrap classes have been initialized by a first task executed by the multitasking virtual machine;

whereby the binary variable indicates to the multitasking virtual machine whether all bootstrap classes have been initialized.

35. (Original) The apparatus of claim 34, further comprising:

an examining mechanism that is configured to note the holder of the class initialization barrier; and

wherein the assigning mechanism is further configured to assigne the holder to the initializer field only if the binary variable is zero.

36. (Original) The apparatus of claim 33, wherein the assigning mechanism is further configured to assign the initializer field of the class to a constant value, wherein the constant value is distinguishable from all other possible values for the initializer field.

37. (Original) The apparatus of claim 36, wherein the constant value is a NULL pointer.

38. (Original) The apparatus of claim 33, wherein a pointer to a runtime data structure representing the shared part of the class is assigned to the initializer field of the class to indicate that the class is the bootstrap class whose initialization is not triggered by a class initialization barrier.

39. (Original) The apparatus of claim 33, further comprising a setting mechanism that is configured to set the class to a fully initialized state upon the class being loading by the task; and

wherein the assigning mechanism is further configured to assign the initializer field of the class to a pointer to a runtime data structure representing the shared part of the class.

40. (Original) The apparatus of claim 33, further comprising an instructing mechanism that is configured to instruct the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the class targeted by the class initialization barrier is equal to the class that defines the program method being compiled.

41. (Original) The apparatus of claim 40, wherein the instructing mechanism is further configured to instruct the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the class targeted by the class initialization barrier is a superclass of the class that defines the program method being compiled.

42. (Original) The apparatus of claim 41, wherein the value of the holder of the class initialization barrier is a pointer to a runtime data structure representing the shared part of the class that defines the program method that holds the class initialization barrier.

43. (Original) The apparatus of claim 42, wherein the instructing mechanism is further configured to instruct the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the value of the initializer field of the class targeted by the class initialization barrier is:

different from the value that indicates that the class is not bootstrap class, and

different from the pointer to the runtime data structure representing the shared part of the class that defines the program method being compiled.

44. (Original) The apparatus of claim 41, wherein the value of the holder of the class initialization barrier is a pointer to the shared runtime data structure representing the program method that holds the class initialization barrier.

45. (Original) The apparatus of claim 44,wherein the instructing mechanism is further configured to instruct the dynamic compiler not to generate native code for the class initialization barrier of the program method being compiled if the value of the initializer field of the class targeted by the class initialization barrier is:

different from the value that indicate that the class is not the bootstrap class, and

different from the pointer to the shared runtime data structure representing the program method being compiled.